MATHEMATICS & MODELING LIBRARY REPORT

BOEING COMPUTER SERVICES COMPANY MM-LR-29

EISPACK User's Guide

H. D. Simon

October 1983

Engineering Technology Applications Division
G-7500, M/S 9C-01
P.O. Box 24346
Seattle, Washington 98124

TABLE OF CONTENTS

TABLE OF CONTENTS (cont.)

1.0 INTRODUCTION

1.1 Purpose

This document describes the structure and usage of EISPACK, a collection of Fortran subroutines for the computation of eigenvalues and eigenvectors of matrices. EISPACK was developed at Argonne National Laboratory, and represents the collective effort of many numerical analysts. The subroutines are considered to be the state-of-the-art in general purpose eigenanalysis software and are widely used in industrial, scientific, and engineering applications.

For the users of the MAINSTREAM-EKS/VSP Cyber and Cray computer systems and of the IBM MVS/TSO computer systems the subroutines in EISPACK are available as one of the "Speciality Mathematical Libraries" supported by BCS. Furthermore, some of the most frequently used EISPACK routines are also integrated (under different names) into the linear algebra section of BCSLIB, the general purpose mathematical library on most BCS facilities (including computers such as PDP and VAX, on which EISPACK itself is not available). Finally eigenvalue problems can also be solved interactively using the EISPACK routines embedded in the matrix computations laboratory MATLAB on the EKS CYBER computers.

Because of the richness of the problem and of the variety of available EISPACK options, one of the main purposes of this document is to guide a potential user to the correct choice of subroutines for a particular eigenvalue problem. In addition, some mathematical background information about eigenvalue problems and their numerical solution is provided, as well as the necessary details on how to access EISPACK on various BCS systems. A prospective user of eigenvalue software should be aware that some understanding of the mathematical background material is necessary to apply EISPACK successfully.

1.2 How to Use this Guide

Several types of questions naturally arise during the numerical solution of an eigenvalue problem. The main body of this report consists of five chapters, each one giving assistance with a particular class of questions:

Chapter 2 : What type of eigenvalue problem is to be solved?
(e.g., symmetric, unsymmetric, generalized, etc.)

Chapter 3 : What computational task is to be achieved?
(e.g., all eigenvalues, some eigenvectors, etc.)

Chapter 4 : Which subroutine should be chosen for the given problem and task?

Chapter 5 : How can this routine be accessed on BCS facilities?
What is the calling sequence?
How can documentation be obtained?

Chapter 6 : What do the results mean?
(e.g., error flags, how accurate are the results?)

Since these are the most common questions a user may ask, it is hoped that this guide is essentially self-contained. For those encountering difficulties beyond the scope of this report, a consultation service is provided (see section 1.3.2). Some extra references are listed in section 1.3.1.

The casual user - someone encountering EISPACK for the first time, or someone who wants a quick and easy solution to a matrix eigenvalue problem, may skip over some sections, depending on background and experience, and consult only those sections relevant to the situation. For example, a user familiar with eigenvalue problems in general, but unfamiliar with EISPACK, may go immediately to Chapter 4.

An option available to the novice user is Appendix A, which is a programmed tutorial that guides the reader through a step by step solution of the eigenvalue problem.

Finally, for the sophisticated user and for those who need extra information, the next section lists available help beyond this report. Users who encounter some unforseen difficulty in solving eigenvalue problems may want to utilize some of these other resources.

1.3 Further Information

1.3.1 References

(see also Appendix B)

The complete EISPACK documentation can be found in two reference books:

Smith et.al.	Matrix Eigensystem Routines - EISPACK Guide	[1]
Garbow et al.	Matrix Eigensystem Routines - EISPACK Extension	[2]

These references are not readily available and may interest only the sophisticated user.

The documentation of the EISPACK based subroutines in BCSLIB is available in the

BCSLIB User's Manual [3].

Information of general interest for users of mathematical software can be found in the

Math/Stat Software Newsletter [4].

Current and back issues of the Newsletter can be obtained from the Newsletter editor at (206) 575-5113. Newsletter issues with relevant articles are:

Vol.7, No.3 :	MATLAB
Vol.7, No.1 :	BCSLIB and Mathematical Software Libraries
Vol.3, No.1 :	EISPACK
Vol.2, No.2 :	"Row Dimension" in Matrix Algebra Software.

1.3.2 Consultation Services

BCS maintains a staff of experts to provide a consultation service for its software users. Help is provided on which routines are available, on appropriate mathematical techniques for a particular problem, and on software usage difficulties encountered by a user. Those needing such advice should contact their local BCS consultation service who will provide assistance and, where necessary, put them in contact with the appropriate specialist. Alternatively, the consultation service listed below may be contacted directly.

For documentation on the Speciality Mathematics Libraries (including EISPACK), contact the Math/Stat Newsletter editor at (206) 575-5113.

For consultation on usage of BCSLIB or the Speciality Mathematics Libraries (including EISPACK) contact the Math/Stat Libraries consultation service at (206) 575-5078.

1.3.3 On-Line Information (General)

Summary information on mathematical, statistical, and utility libraries and packages, consultation services, and the quarterly Newsletter is available on-line. It is approximately 15 pages long and is updated regularly to reflect software or service changes. Users of EISPACK are encouraged to obtain a personal copy on a regular basis to be informed about recent changes.

On the EKS CYBER system a copy can be obtained by the following commands:

```
GET, MTHINFO/UN=EKSAPP.
ROUTE,MTHINFO,DC=PR,MB=mailbox,UN=RJE user number.
```

On the TSO systems use the following JCL statement:

```
//EXEC MTHINFO.
```

2.0 MATHEMATICAL BACKGROUND

2.1 General Remarks

Notation. In this report the following notation is adopted. Column vectors are denoted by lower case bold letters like $\mathbf{x},\mathbf{y},\mathbf{z}$. The components of $\mathbf{x}$ are real or complex numbers x_j, $j=1,2, \ldots n$, where n is the dimension of $\mathbf{x}$. Upper case letters like A, B, C denote matrices and Greek letters like λ, α denote scalars. The entries of the matrix A are given by a_{ij}, i.e. $A=(a_{ij})$, $i,j = 1,2, \ldots n$. The transpose of A is given by A^T, and the complex conjugate transpose of a matrix A or a vector $\mathbf{x}$ is denoted by A^* or $\mathbf{x}^*$, respectively. The scalar quantity $||\mathbf{x}|| = \sqrt{\mathbf{x}^*\mathbf{x}}$ is the ordinary Euclidean norm of $\mathbf{x}$.

This chapter discusses the various forms in which matrix eigenvalue problems can be encountered. Certain properties of the coefficient matrix (or matrices) imply certain other properties of the eigenvalues and vectors, which can make their numerical computation easier and more robust. EISPACK subroutines take advantage of these special matrix properties, when appropriate, and therefore guarantee more reliable and/or faster results. It is therefore to the advantage of the user to be aware of any special features of his eigenvalue problem. The purpose of this chapter is to discuss the various forms of the eigenvalue problem, the corresponding mathematical properties of the eigenvalues and vectors, and how these special properties are exploited by the algorithms in EISPACK.

Eigenvalue problems can be stated in several different forms. The most common one is: given an n by n matrix A, find scalars λ and vectors $\mathbf{x} \neq 0$ such that

$$A\,\mathbf{x} = \lambda\,\mathbf{x}\,. \qquad (1)$$

Equation (1) will be refered to as the standard eigenvalue problem. The standard eigenvalue problem is discussed in more detail in section 2.2.

A more complicated form is the so called generalized eigenvalue problem: given n by n matrices A and B find scalars λ and vectors $\mathbf{x} \neq 0$ such that

$$A\,\mathbf{x} = \lambda\,B\,\mathbf{x}\,. \qquad (2a)$$

Related to the generalized eigenvalue problem are the AB eigenproblem:

$$A\,B\,\mathbf{x} = \lambda\,\mathbf{x}\ , \qquad (2b)$$

and the BA eigenproblem

$$B\,A\,\mathbf{x} = \lambda\,\mathbf{x}\ . \qquad (2c)$$

These three problems are discussed in section 2.3. A naive approach to problems (2abc) would be to form the matrices $B^{-1}A$, AB, or BA explicitly and then regard the problem as a standard eigenproblem. This, however, is sometimes difficult, and may even be impossible in the case (2a). It is unwise in cases (2bc) if the matrices satisfy certain special properties which can be utilized by the algorithms in EISPACK.

There are some other forms of the eigenvalue problem which are occasionally encountered, like the quadratic eigenvalue problems or λ-matrix problems. Although there are no EISPACK routines which are explicitly designed for the solution of these problems, some suggestions for their solution will be made in section 2.4.

In section 2.5 the singular value decomposition of a matrix will be discussed. For some applications the singular values and vectors instead of the eigenvalues/vectors are needed. Examples are the numerical determination of the rank of a matrix, the pseudo-inverse of a rectangular matrix, or the solution of overdetermined linear systems. Finally, section 2.6 gives an overview of the type of numerical methods used by EISPACK.

For a first reading it is suggested that the casual user of this guide proceed as follows:

- If the problem is a standard eigenvalue problem, read section 2.2 (and maybe section 2.6). Then continue with Chapter 3.
- If the problem is a generalized, an AB, or a BA problem read sections 2.2 and 2.3 (and maybe 2.6). Then continue with Chapter 3.
- If the problem is none of the above, then read sections 2.4 and 2.5, and/or ask the consultants for help.

2.2 The Standard Eigenvalue Problem.

2.2.1 Mathematically Different Forms of the Standard Eigenvalue Problem.

As mentioned in the previous section the standard eigenvalue problem is stated as follows: given an n by n real (or complex) matrix A, find scalars λ and vectors $\mathbf{x} \neq 0$ such that

$$A \mathbf{x} = \lambda \mathbf{x} . \qquad (1)$$

A scalar λ which satisfies (1) is called an eigenvalue of A and the corresponding vector $\mathbf{x}$ is called an eigenvector of A. The pair λ, $\mathbf{x}$ is sometimes referred to as an eigenpair. By convention $\mathbf{x}=0$ is not considered to be an eigenvector. Also note that if $\mathbf{x}\neq 0$ is an eigenvector, then $\alpha\mathbf{x}$ for an arbitrary scalar α is also an eigenvector. If one talks about the eigenvector, it is usually assumed that $\mathbf{x}$ is normalized in some way, for example $||\mathbf{x}|| = 1$.

If the eigenvalue problem is stated as above and nothing else is known about A the problem is called the real general (or complex general) eigenvalue problem. It is important to keep the following facts about the real (or complex) general eigenvalue problem in mind:

1) There are always n eigenvalues λ_1, λ_2, ... λ_n, which may not be distinct.
2) The eigenvalues may be complex. (This is true even if A is real. Then the eigenvalues occur in complex conjugate pairs.)
3) There may not be n different eigenvectors. (Even though there are n eigenvalues).
4) Even if there is a set of n eigenvectors, some of them may be arbitrarily close to each other, and consequently hard to distinguish from each other numerically.

These mathematical facts have some important consequences for the practical computation of eigenvalues/vectors in the real (or complex) general case:

- Even for the real general eigenvalue problem one should expect to obtain complex answers.
- It may be difficult or impossible to compute a full set of eigenvectors.

All these difficulties disappear if the matrix A has one special property. The real matrix A is called symmetric, if $A = A^T$, i.e. if $a_{ij} = a_{ji}$ for i,j = 1, 2 ... n. Correspondingly a complex matrix A is called Hermitian, if $A = A^*$, i.e. $a_{ij}=\bar{a}_{ji}$ for i,j = 1, 2, ... n. Important facts about the real symmetric (or the complex Hermitian) eigenvalue problem are:

1) There are always n eigenvalues $\lambda_1, \lambda_2, \ldots \lambda_n$, which may not be distinct.
2) The eigenvalues are all real. (This is true even if A is complex Hermitian.)
3) There are always n different eigenvectors.
4) Eigenvectors x, y corresponding to different eigenvalues are orthogonal, i.e. $x^Ty = 0$ (or $x^*y = 0$ in the complex Hermitiam case). Consequently, they are easy to distinguish numerically.

These facts about the real symmetric (or complex Hermitian) eigenvalue problem make the actual computation of eigenvalues/vectors much easier as compared to the real (or complex) general case. Correspondingly, there are different algorithms which are appropriate. Obviously the algorithm for the general case will also work for the symmetric (or Hermitian) case, but it is to the advantage of the user to use the more specialized and therefore more efficient algorithm.

The distinction between symmetric and nonsymmetric matrices is of central importance for eigenvalue calculations. Symmetry should be always utilized in eigenvalue computations, because it guarantees better results (see section 6.3). This distinction is therefore much more important than the distinctions which will be made in the next subsection.

2.2.2 Different Ways of Storing the Coefficient Matrix

In this subsection the various possibilities of how to store the coefficient matrix will be discussed. The differences between the various forms concern only the implementation. The mathematical properties of the eigenvalues/eigenvectors are not affected by these different forms of the eigenvalue problem. The distinctions here are made because of the structure of EISPACK, but not because of mathematical necessities.

The symmetry of A can also be utilized in order to save storage. It is only necessary to store the $n*(n+1)/2$ entries of the lower triangular part of A. They can be stored in a one dimensional array by rows in the order a_{11}, a_{21}, a_{22}, a_{31}, a_{32}, a_{33}, a_{41},... . This representation of the real symmetric matrix A will be called real symmetric packed.

A similar idea can be used if A is an n by n complex Hermitian matrix. Each complex entry can be represented as a pair of reals as follows: $a_{ij} = (b_{ij},c_{ij})$. If the real parts of the entries of the lower triangular part of A are stored in the lower triangle of a two dimensional array, and if the corresponding imaginary parts are stored in the upper triangle, then A will be called complex Hermitian packed. For example if n=4 the matrix A is stored as follows, using the usual columnwise storage notation for doubly subscripted arrays:

$$\begin{array}{cccc} b_{11} & c_{21} & c_{31} & c_{41} \\ b_{21} & b_{22} & c_{32} & c_{42} \\ b_{31} & b_{32} & b_{33} & c_{43} \\ b_{41} & b_{42} & b_{43} & b_{44} \end{array}$$

Note that the imaginary parts of the diagonal entries are zero, and are not stored explicitly.

Certain subroutines in EISPACK are tailored to this packed storage mode of matrices. The user who is concerned about storage limitations may want to consider these routines instead of the more general ones.

The same can be said about <u>real symmetric band matrices.</u> If A is a real symmetric matrix of order n, then it is said to be banded if there is an integer k, $(0 \leq k < n)$ such that $a_{ij} = 0$ whenever $|i - j| > k$. For example, when n=5 and k=2, the matrix A has the following form:

$$\begin{bmatrix} a_{11} & a_{21} & a_{31} & 0 & 0 \\ a_{21} & a_{22} & a_{32} & a_{42} & 0 \\ a_{31} & a_{32} & a_{33} & a_{43} & a_{53} \\ 0 & a_{42} & a_{43} & a_{44} & a_{54} \\ 0 & 0 & a_{53} & a_{54} & a_{55} \end{bmatrix}$$

For symmetric banded matrices, storage is saved by storing only the lower triangle of A in a two dimensional array of order n*(k+1). The matrix A above is then stored as follows, where (.) stands for an arbitrary value:

$$\begin{matrix} (.) & (.) & a_{11} \\ (.) & a_{21} & a_{22} \\ a_{31} & a_{32} & a_{33} \\ a_{42} & a_{43} & a_{44} \\ a_{53} & a_{54} & a_{55} \end{matrix}$$

EISPACK does not provide routines for the corresponding banded complex Hermitian case. However, it does provide routines for an important class of banded symmetric matrices. These are the <u>real symmetric tridiagonal matrices</u>, i.e. banded matrices with k=1. For example the symmetric tridiagonal matrix corresponding to n=5 is given by

$$\begin{bmatrix} a_{11} & a_{21} & 0 & 0 & 0 \\ a_{21} & a_{22} & a_{32} & 0 & 0 \\ 0 & a_{32} & a_{33} & a_{43} & 0 \\ 0 & 0 & a_{43} & a_{44} & a_{54} \\ 0 & 0 & 0 & a_{54} & a_{55} \end{bmatrix}$$

Here the diagonal and subdiagonal elements are stored in two one-dimensional arrays. Again, there is no special subroutine for complex Hermitian tridiagonal matrices.

The last class of matrices for which EISPACK provides special routines are sign-symmetric real tridiagonal matrices. These are tridiagonal real matrices whose offdiagonal elements have matching signs, that is , for all i, $\text{sign}(a_{i,i+1}) = \text{sign}(a_{i+1,i})$. Although these matrices are not symmetric, the properties of their eigenvalues and eigenvectors are the same as listed above for real symmetric matrices.

At this point it is appropriate to summarize the above classification scheme for the standard eigenvalue problem. The relationship between the various classes is exhibited in the following two tree structures, one for real matrices and one for complex matrices.

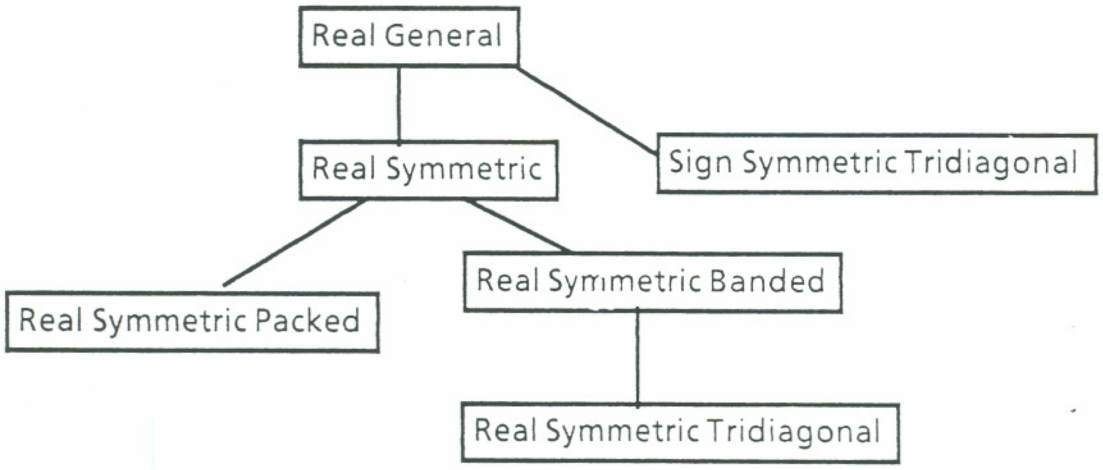

Figure 2.1. Classes of Matrices for the Real Eigenvalue Problem.

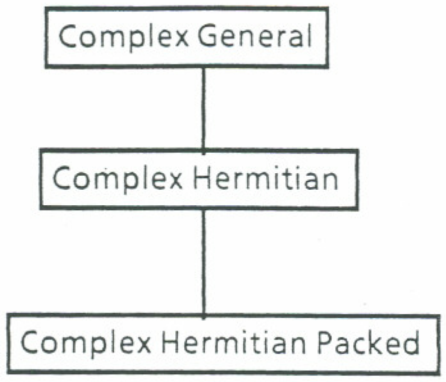

Figure 2.2. Classes of Matrices for the Complex Eigenvalue Problem.

Each box in the two trees corresponds to a class of matrices for which EISPACK provides the implementation of a special algorithm. At this point the "casual" user with a problem to solve is encouraged to locate the corresponding box which contains the right algorithm. There are several facts to remember about choosing the "right box":

- The classes at the lower level of the tree are always completely included in the classes above, which means, for example, that a real symmetric tridiagonal problem can always be solved using the algorithm for real symmetric matrices.
- However, in terms of efficiency and reliability it pays to choose the box which most closely fits the given problem. The reader is especially urged to make use of any symmetry in the problem.

2.3 The Generalized Eigenvalue Problem

This section discusses generalizations of the standard eigenvalue problem involving two real n by n matrices A and B. The generalized eigenvalue problem has the form (c.f. section 2.1):

$$A \mathbf{x} = \lambda B \mathbf{x} . \qquad (2a)$$

EISPACK provides an algorithm which solves the generalized eigenvalue problem in its original form (2a). This algorithm is more efficient than the alternative of reformulating (2a) to

$$B^{-1}A \mathbf{x} = \lambda \mathbf{x} ,$$

and then solving the standard eigenvalue problem for $B^{-1}A$ with one of the methods from section 2.2. Besides being inefficient, forming $B^{-1}A$ may be inaccurate or even impossible, if B is ill-conditioned or singular. For these reasons forming $B^{-1}A$ is not recommended.

The eigensystem of (2a) has the same features as the eigensystem of (1) in the real general case (see section 2.2), but the following additional cases may occur:

- Vectors $\mathbf{z}$, with $\mathbf{z} \neq 0$ and with $B\mathbf{z} = 0$ are regarded as eigenvectors. The corresponding eigenvalue is considered to be infinite.
- If there is an eigenvector $\mathbf{z} \neq 0$ with $A\mathbf{z} = 0$ and $B\mathbf{z} = 0$ then every real number is an eigenvalue correponding to that vector.

The algorithm used in EISPACK is able to detect these special cases by computing pairs of scalars α and β which satisfy

$$\beta A \mathbf{x} = \alpha B \mathbf{x} .$$

A careful examination of the ratios $\lambda = \alpha/\beta$ may then reveal the special cases (for more details see section 3.3.1 in [2]).

Fortunately in many practical applications the matrices A and B are real symmetric matrices, and B is positive definite. (A matrix B is called positive definite if all its eigenvalues are greater than zero). The

2.3 The Generalized Eigenvalue Problem

This section discusses generalizations of the standard eigenvalue problem involving two real n by n matrices A and B. The <u>generalized eigenvalue problem</u> has the form (c.f. section 2.1):

$$A\,\mathbf{x} = \lambda\,B\,\mathbf{x}\,. \qquad (2a)$$

EISPACK provides an algorithm which solves the generalized eigenvalue problem in its original form (2a). This algorithm is more efficient than the alternative of reformulating (2a) to

$$B^{-1}A\,\mathbf{x} = \lambda\ \mathbf{x}\,,$$

and then solving the standard eigenvalue problem for $B^{-1}A$ with one of the methods from section 2.2. Besides being inefficient, forming $B^{-1}A$ may be inaccurate or even impossible, if B is ill-conditioned or singular. For these reasons forming $B^{-1}A$ is not recommended.

The eigensystem of (2a) has the same features as the eigensystem of (1) in the real general case (see section 2.2), but the following additional cases may occur:

- Vectors $\mathbf{z}$, with $\mathbf{z} \neq 0$ and with $B\mathbf{z} = 0$ are regarded as eigenvectors. The corresponding eigenvalue is considered to be infinite.
- If there is an eigenvector $\mathbf{z} \neq 0$ with $A\mathbf{z} = 0$ and $B\mathbf{z} = 0$ then every real number is an eigenvalue correponding to that vector.

The algorithm used in EISPACK is able to detect these special cases by computing pairs of scalars α and β which satisfy

$$\beta\,A\,\mathbf{x} = \alpha\,B\,\mathbf{x}\,.$$

A careful examination of the ratios $\lambda = \alpha/\beta$ may then reveal the special cases (for more details see section 3.3.1 in [2]).

Fortunately in many practical applications the matrices A and B are real symmetric matrices, and B is positive definite. (A matrix B is called positive definite if all its eigenvalues are greater than zero). The resulting real generalized symmetric eigenvalue problem has properties corresponding to the real symmetric eigenproblem discussed in section 2.2. There are n real eigenvalues and n corresponding linearly independent eigenvectors. Pairs of different eigenvectors $\mathbf{x},\mathbf{y}$ are no longer orthogonal in the ordinary sense, however it does hold that $\mathbf{x}^T B \mathbf{y} = 0$. Whatever has been said about the relationship between real general and real symmetric eigenvalue problems carries over correspondingly to the real generalized and the real generalized symmetric eigenvalue problem

There are two additional less common forms for the symmetric generalized eigenvalue problem which can be solved directly by EISPACK, namely the problems

$$AB\,\mathbf{x} = \lambda\,\mathbf{x}\ , \qquad (2b)$$

$$BA\,\mathbf{x} = \lambda\,\mathbf{x}\ , \qquad (2c)$$

where A and B are real symmetric and B is positive definite. These two problems, refered to as the AB (or BA) eigenvalueproblem, have the same properties as the generalized symmetric eigenvalue problem.

The basic four generalized eigenvalue problems for which there are solution algorithms provided in EISPACK are summarized in the table below.

Problem	Name	Matrix A	Matrix B
$A\mathbf{x}=\lambda B\mathbf{x}$	Real generalized	real	real
$A\mathbf{x}=\lambda B\mathbf{x}$	Real generalized symmetric	symmetric	positive definite
$AB\mathbf{x}=\lambda\mathbf{x}$	AB	symmetric	positive definite
$BA\mathbf{x}=\lambda\mathbf{x}$	BA	symmetric	positive definite

Table 2.1. Classification of the Generalized Eigenvalue Problems.

2.4 Quadratic Eigenvalue Problems and λ-Matrix Problems

(see also [2], section 3.3.5)

In some applications, quadratic eigenvalue problems of the form

$$(\lambda^2 C_0 + \lambda C_1 + C_2)\,\mathbf{x} = 0 \tag{3}$$

arise, where C_0, C_1, and C_2 are real n by n matrices. The scalars λ, which make the matrix in parentheses singular, and possibly also the corresponding null vectors $\mathbf{x}$ are to be found. This problem can be solved by forming the 2n by 2n matrices

$$A = \begin{bmatrix} -C_1 & -C_2 \\ I & 0 \end{bmatrix} \qquad B = \begin{bmatrix} C_0 & 0 \\ 0 & I \end{bmatrix} .$$

The problem (3) is then equivalent to the generalized real eigenvalue problem (see section 2.3)

$$A\,\mathbf{z} = \lambda\,B\,\mathbf{z} \quad ,$$

where $\mathbf{z} = \begin{bmatrix} \lambda x \\ x \end{bmatrix}$. This can be solved using the methods for (2a).

More generally a λ-matrix problem is given by

$$C(\lambda)\,\mathbf{x} = 0 \quad , \tag{4}$$

where each entry of the real n by n matrix $C(\lambda)$ is a function of the parameter λ. Solutions to (4) are values λ, for which $C(\lambda)$ is singular, together with the corresponding nullvectors. If $C(\lambda) = \lambda^r C_0 + \lambda^{r-1} C_1 + \ldots + \lambda C_{r-1} + C_r$ with constant matrices C_i, i=1,...r, the problem (4) can be reduced to a generalized eigenvalue problem by the same method described above. Numerical algorithms for more complicated functional dependencies of the entries of $C(\lambda)$ require consultation.

2.5 The Singular Value Decomposition

(adapted from [2], section 2.4.1)

The singular value decomposition (SVD) proceeds from a real rectangular matrix A with m rows and n columns. Although m and n are not constrained, the primary applications of the decomposition have $m \geq n$. The decomposition is commonly written

$$A = U \Sigma V^T \qquad (5)$$

where Σ is an n by n diagonal matrix with entries $\sigma_j \geq 0$, j=1, 2,....n, and U and V are matrices with orthonormal columns of dimensions m by n and n by n respectively. Note that U has the same dimensions as A and that V has as many columns as A but is square. The elements of Σ are called the singular values of A, and the columns of U and V are the left and right singular vectors, respectively.

Theoretically the SVD can be characterized by the fact that the singular values are the square roots of the eigenvalues of A^TA, the columns of V are the corresponding eigenvectors, and the columns of U are certain eigenvectors of AA^T. However this is not a satisfactory basis for computation, because roundoff errors in the formation of A^TA often destroy pertinent information.

One fundamental application of the singular value decomposition is the determination of the rank of a matrix. Since the orthogonal transformations U and V preserve linear independence, the ranks of A and Σ are the same, and the rank of A can be found by counting the number of singular values greater than zero. In a practical computation, however, it is likely that even for a rank deficient matrix none of the singular values will be exactly zero. Therefore one has to introduce a threshold τ , which will depend on the problem at hand, and regard all singular values less than τ as effectively zero. For the choice of τ and for other applications of the SVD see [1], section 2.4.1.

2.6 Algorithms

(adapted from [5])

This is not the place to describe all the algorithms employed by EISPACK, nor to assess their relative performance. Here only a little mathematical background will be presented, and two examples will be shown which illustrate some of the more frequently used algorithms.

Almost all the algorithms used in EISPACK are based on similarity transformations. Two matrices A and B are similar if there is a nonsingular matrix S for which

$$B = S^{-1}AS.$$

The transformation $S^{-1}AS$ is called a similarity transformation. Similar matrices have the same eigenvalues, and the eigenvectors of one are easily obtained from the other. If B were diagonal, then its diagonal elements would be the eigenvalues of A. Moreover the columns of S would be the eigenvectors of A.

It is not necessary, and numerically it may not even be desirable to completely diagonalize a matrix. If B is merely triangular, then the eigenvalues are still on the diagonal and the eigenvectors can be computed by a fairly straightforward substitution process.

A real matrix S is called orthogonal if $S^TS = I$ or, equivalently, $S^T = S^{-1}$. A complex matrix S is called unitary if these same conditions hold with S^T replaced by S^*. Similarity transformations based on orthogonal and unitary matrices are particularly attractive from a numerical point of view because they do not magnify any errors that may be present in the input data, or that may be introduced during the computation. For this reason numerical linear algebra is fortunate to have available the following theorem due to Schur: any matrix can be triangularized by a unitary similarity transformation.

Most of the techniques employed in EISPACK utilize variants of Schur's theorem. It is usually not possible to compute Schur's transformation with a finite number of rational arithmetic operations. Instead the algorithms employ a potentially infinite sequence of similarity transformations

$$A_{k+1} = S^{-1}{}_k A_k S_k$$

for which A_k approaches an upper triangular matrix. The sequence is terminated when all of the subdiagonal elements of a particular matrix A_k are less than the roundoff errors involved in the computation. These elements can then be set to zero without introducing any more perturbations in the eigenvalues than have already been caused by the preceeding transformations. The diagonal elements of the resulting matrix A_k are then the desired approximations to the eigenvalues of the original matrix. The corresponding eigenvectors can be readily computed if they have been requested.

For reasons pointed out in section 2.2 it is important to have special algorithms which deal with symmetric matrices. The only similarity transformations which also preserve symmetry are those based on orthogonal matrices. The following example outlines the basic steps used in the algorithm employed by EISPACK. The input matrix is:

$$\begin{bmatrix} 5 & 4 & 3 & 2 & 1 \\ 4 & 5 & 4 & 3 & 2 \\ 3 & 4 & 5 & 4 & 3 \\ 2 & 3 & 4 & 5 & 4 \\ 1 & 2 & 3 & 4 & 5 \end{bmatrix}$$

The initial orthogonal similarity transformations which are carried out reduce the matrix to tridiagonal form. An n by n matrix requires n-2 such transformations, each of which introduces zeros into a particular row or column of the matrix, while preserving the symmetry and preserving the zeros introduced by the previous transformations. In the case of the 5 by 5 example above, the result of the first transformation is a matrix which has three zeros in the last row and column. The next transformation introduces two more zeros into the fourth row and column. The final transformation places one more zero in the third row and column. One obtains this tridiagonal matrix:

$$\begin{bmatrix} 0.6594 & -0.1438 & 0 & 0 & 0 \\ -0.1438 & 0.9687 & 0.5678 & 0 & 0 \\ 0 & 0.5678 & 5.3052 & 4.4192 & 0 \\ 0 & 0 & 4.4192 & 13.0667 & -5.4772 \\ 0 & 0 & 0 & -5.4772 & 5.0000 \end{bmatrix}$$

Since the result of the initial reduction is a symmetric tridiagonal matrix, it can be stored in just two vectors - one with n components for the diagonal and one with n-1 components for the offdiagonal.

EISPACK includes several routines for computing the eigenvalues of a real, symmetric tridiagonal matrix. Many, but not all of these routines are variants of the QR algorithm, originally published by Francis in 1961 and perfected by Wilkinson and Reinsch in [6].

The symmetric, tridiagonal QR algorithm produces a sequence of similar matrices whose offdiagonal elements are decreasing in magnitude and whose diagonal elements are approaching the desired eigenvalues. With the variant of the algorithm used here the intention is to reduce the first offdiagonal element most rapidly. After three similarity transformations one obtains:

$$\begin{bmatrix} 0.5484 & 0.0000 & 0 & 0 & 0 \\ 0.0000 & 0.7641 & 0.0085 & 0 & 0 \\ 0 & 0.0085 & 1.2737 & 0.0167 & 0 \\ 0 & 0 & 0.0167 & 5.2501 & -0.4103 \\ 0 & 0 & 0 & -0.4103 & 17.1637 \end{bmatrix}$$

Notice that all the offdiagonal elements have generally decreased in size, and that the first offdiagonal element has decreased a geat deal. It has now reached a size which is comparable to the roundoff errors made during the calculation, and so setting it to zero can be regarded simply as another roundoff error. The first diagonal element is now an accurate approximation to one of the eigenvalues of the original matrix.

The next two iterations involve only the 4 by 4 submatrix. One obtains after two iterations:

$$\begin{bmatrix} 0.5484 & 0 & 0 & 0 & 0 \\ 0 & 0.7639 & -0.0000 & 0 & 0 \\ 0 & -0.0000 & 1.2738 & 0.0003 & 0 \\ 0 & 0 & 0.0003 & 5.2362 & -0.0315 \\ 0 & 0 & 0 & -0.0315 & 17.1777 \end{bmatrix}$$

The second offdiagonal element is now negligible and the second diagonal element approaches another eigenvalue. Three more iterations are needed to reduce the remaining offdiagonal elements to roundoff level. The final diagonal matrix is

$$\begin{bmatrix} 0.5484 & 0 & 0 & 0 & 0 \\ 0 & 0.7639 & 0 & 0 & 0 \\ 0 & 0 & 1.2738 & 0 & 0 \\ 0 & 0 & 0 & 5.2361 & 0 \\ 0 & 0 & 0 & 0 & 17.1776 \end{bmatrix}$$

In this example a total of 15 similarity transformations were required, 3 for the initial reduction and 12 for the QR iterations. The later transformations involved considerably less arithmetic than the earlier ones because they were done on matrices with more zero elements.

Let S denote the product of all the orthogonal similarity transformations which were required, and let D denote the final diagonal matrix. Then $S^{-1}AS = D$, and hence $AS = SD$. This shows that the columns of S are the eigenvectors of A. Moreover, since S is the product of orthogonal matrices it must be also orthogonal.

Nonsymmetric matrices involve somewhat different techniques, although the general approach of an initial reduction followed by some QR type iteration is still followed. Since there is no symmetry to be preserved, similarity transformations can be based on nonorthogonal matrices. Algorithms which employ elimination methods require less arithmetic, and hence are potentially faster, than those which use orthogonal transformations. However, such algorithms may produce somewhat less accurate results and, in extreme, contrived examples, may be completely unstable. Deciding between these two classes of algorithms involves tradeoffs between execution speed and numerical reliability. When designing a general purpose library, such decisions are difficult to make, and EISPACK therefore contains implementations of both algorithms.

The following example illustrates the behavior of these algortihms for a real general matrix. The input matrix is

$$\begin{bmatrix} 5 & 4 & 3 & 2 & 1 \\ 9 & 5 & 4 & 3 & 2 \\ 8 & 9 & 5 & 4 & 3 \\ 7 & 8 & 9 & 5 & 4 \\ 6 & 7 & 8 & 9 & 5 \end{bmatrix}$$

The first step is to use a subroutine to produce the following <u>Hessenberg matrix:</u>

$$\begin{bmatrix} 5.0000 & 8.8889 & 4.0174 & 4.6055 & 2.0000 \\ 9.0000 & 12.2222 & 6.2902 & 6.4082 & 3.0000 \\ 0 & 16.2963 & 11.7891 & 10.9525 & 7.0000 \\ 0 & 0 & -2.5925 & -2.6545 & -1.9705 \\ 0 & 0 & 0 & 1.3901 & -1.3569 \end{bmatrix}$$

A matrix $A = (a_{ij})$ is called an (upper) Hessenberg matrix if $a_{ij}=0$ for all $i>j+1$. As in the symmetric case the reduction to tridiagonal form, here the reduction to Hessenberg form can be carried out in finitely many steps. The application of the QR algorithm to a matrix in Hessenberg form can then be carried out more efficiently.

In the example above there was very little growth in the size of the elements during the initial reduction, so there was very little roundoff error magnification. The next step is to carry out a variant of the QR iteration. After a few iterates one obtains:

$$\begin{bmatrix} 24.7514 & -11.1821 & -3.6155 & 12.9960 & 7.0641 \\ 0 & 3.6275 & -1.3330 & 4.9235 & 6.0036 \\ 0 & 0 & -1.2203 & -1.8985 & -0.4090 \\ 0 & 0 & 1.2977 & -0.6818 & 2.4406 \\ 0 & 0 & 0 & 0 & -1.4768 \end{bmatrix}$$

Three of the eigenvalues are revealed in diagonal positions 1, 2, and 5. The 2 by 2 submatrix that includes diagonal positions 3 and 4 is the source of a pair of complex conjugate eigenvalues, $-0.9511 \pm 1.5463i$. The entire computation is done with real arithmetic, even though the final results are complex.

3.0 Solution Options

There are many possible requirements when solving the eigenvalue problem, for example: compute all eigenvalues, compute only one eigenvalue and the corresponding eigenvector, compute some eigenvalues and the corresponding eigenvectors, etc. One way of providing for all these different options would be to always compute the complete eigensystem, and then extract the desired quantities. For many problems this may be very inefficient and wasteful. This short chapter describes the different solution options provided by EISPACK, together with some guidelines for chosing the proper option.

The standard options for the solution are:

A) All eigenvalues

B) All eigenvalues and some selected eigenvectors

C) All eigenvalues and all eigenvectors

D) The largest or smallest eigenvalues

E) The largest or smallest eigenvalues and the corresponding eigenvectors

F) The eigenvalues in a specified interval

G) The eigenvalues in a specified interval and the corresponding eigenvectors.

The solution options D-G apply only to symmetric (Hermitian, generalized symmetric) eigenproblems.

These are all the capabilities provided by EISPACK. There are obviously many computational tasks which arise in practice and for which there are no corresponding solution options. Examples of these tasks are: finding all eigenvalues of a real matrix which have a negative real part, or finding the eigenvalue of a real matrix with largest (smallest) real part. All these and many similar tasks can be accomplished by using the appropriate subroutines with more general solution options. In some cases it may be possible to combine or to modify EISPACK routines for these tasks. The user may consult the EISPACK Guide or seek help from the consultants (see section 1.3.2).

One final rule of thumb concerning the choice of various solution options is as follows:

If more than one fourth of the eigenvalues are needed it is usually faster to compute all the eigenvalues instead of using one of the specialized options.

4.0 The EISPACK Subroutines

4.1 General Remarks

The purpose of this chapter is to guide the user to the appropriate subroutine(s). It is assumed that after studying chapters 2 and 3, the user with a specific eigenvalue problem has identified the appropriate problem class and solution option.

There are some considerations to be be addressed before selecting a subroutine. These concern the ways of using EISPACK on BCS computers. The various possibilities are first discussed in section 4.2. The general organization of EISPACK is explained in section 4.3. It is then a fairly straightforward task to find the corresponding EISPACK subroutine. Section 4.4 shows the EISPACK subroutine for each combination of problem class and solution option.

Chapter 4 is concluded by section 4.5 on MATLAB and section 4.6 on EISPAC (note the missing "K"). These are two software products which faciliate the use of EISPACK on the EKS-CYBER or the TSO systems, respectively.

It is suggested that the casual user study Sections 4.2, 4.3, and 4.4, and that Sections 4.5 and 4.6 be read only if needed.

4.2 Eigenvalue Software on BCS Computers

EISPACK based software for solving eigenvalue problems is available in several forms on the different BCS systems. As already mentioned in the introduction, there are

(1) BCSLIB, which includes subroutines derived from EISPACK,
(2) EISPACK itself, which BCS supports as one of its "Speciality Mathematical Libraries",
(3) the EISPAC control program on IBM MVS/TSO systems,
(4) the interactive MATLAB program on EKS CYBER computers.

The main concern of this document is of course EISPACK proper. However, there are situations for which the other forms of the eigenvalue software may be more appropriate.

The scope of the subroutines in BCSLIB is more limited than the scope of those in EISPACK. The BCSLIB subroutines are driver subroutines which call the most frequently used EISPACK subprograms. As such they are almost identical to the corresponding EISPACK drivers to be discussed in the next section. The following table gives an overview of the available subroutines:

Precision	$A\mathbf{x}=\lambda\mathbf{x}$ A general	$A\mathbf{x}=\lambda\mathbf{x}$ A symmetric (A Hermitian)	$A\mathbf{x}=\lambda B\mathbf{x}$ A,B, general	$A\mathbf{x}=\lambda B\mathbf{x}$ A symmetric, B symmetric positive definite
REAL	HSGEEV	HSSYEV	HSGEGV	HSSYGV
DOUBLE	HDGEEV	HDSYEV	HDGEGV	HDSYGV
COMPLEX	HCGEEV	HCHIEV		
COMPLEX*16	HZGEEV	HZHIEV		

Table 4.1. BCSLIB subprograms for the eigenvalue problem.

There are two solution options for each subroutine:

(1) compute all the eigenvalues or

(2) compute all the eigenvalues and eigenvectors.

Every eigenvalue problem from section 2.3 and 2.4 could be solved with these subroutines. There are several good reasons why one should use the BCSLIB routines:

- They are more commonly available. BCSLIB is installed on all major BCS machines , whereas EISPACK is only available on EKS and TSO.
- The BCSLIB routines are more easily accessed. BCSLIB is the default library (except on PDP computers) and automatically searched when a Fortran program is loaded. Thus a call statement is all that is needed to use the BCSLIB subroutines.
- They are just as reliable as their EISPACK counterparts.
- They provide simpler complex matrix storage.
- More comprehensive error reporting by the BCSLIB standard error handler.

For these reasons, use of the BCSLIB routines is recommended. Resorting to EISPACK subroutines is only necessary in the following cases:

- when a special problem has to be solved, one which cannot be treated by the BCSLIB routines, or
- when efficiency is the primary concern of the user, or
- when the user wants to utilize the better acccuracy in the smaller eigenvalues, which is obtained by using certain EISPACK routines.

Since the EISPACK routines are more specialized than the BCSLIB routines, they will probably solve special problems faster and with less storage (see chapters 2 and 3). Also, note that there is a minor difference between BCSLIB and EISPACK in the storage mode for complex matrices and savings in storage may be realized by using the EISPACK routines.

If BCSLIB routines are chosen, it is recommended that the double precision versions be used on IBM, VAX, and PDP systems. The single precision version is sufficient for most scientific applications on CYBER and CRAY systems, and is therefore the only version available on these systems.

4.3 Organization of EISPACK

(adapted from [5])

EISPACK contains 13 drivers, each intended for a different type of matrix. Twelve of the drivers provide two options: compute all the eigenvalues or compute all the eigenvalues and eigenvectors. One of the drivers computes all the eigenvalues and only some of the eigenvectors of a symmetric matrix. Seven of the drivers solve the standard eigenvalue problem involving a single real matrix A. These seven drivers are:

Driver	Problem	Matrix A
RG	$A\mathbf{x}=\lambda\mathbf{x}$	real general
RS	$A\mathbf{x}=\lambda\mathbf{x}$	real symmetric
RSM	$A\mathbf{x}=\lambda\mathbf{x}$	symmetric; all values, some vectors
RSB	$A\mathbf{x}=\lambda\mathbf{x}$	symmetric band
RSP	$A\mathbf{x}=\lambda\mathbf{x}$	symmetric packed
RST	$A\mathbf{x}=\lambda\mathbf{x}$	symmetric tridiagonal
RT	$A\mathbf{x}=\lambda\mathbf{x}$	sign-symmetric tridiagonal

Two of the drivers solve the standard eigenvalue problem for complex matrices:

Driver	Problem	Matrix A
CG	$A\mathbf{x}=\lambda\mathbf{x}$	complex general
CH	$A\mathbf{x}=\lambda\mathbf{x}$	complex Hermitian

Four of the drivers solve generalized eigenvalue problems:

Driver	Problem	Matrix A	Matrix B
RGG	$A\mathbf{x}=\lambda B\mathbf{x}$	real general	real general
RSG	$A\mathbf{x}=\lambda B\mathbf{x}$	symmetric	symmetric, pos.def.
RSGAB	$AB\mathbf{x}=\lambda\mathbf{x}$	symmetric	symmetric, pos.def.
RSGBA	$BA\mathbf{x}=\lambda\mathbf{x}$	symmetric	symmetric, pos.def.

These driver subroutines provide easy access to many of EISPACK's capabilities. The user who is satisfied with these capabilities, and whose problems do not make heavy demands for computer time and storage, need not be concerned with any further details of EISPACK organization.

The drivers are actually just "shell" subroutines which may call as many as four other EISPACK subroutines to do the actual computations. Several of these other subroutines are used by more than one driver. On the other hand some subroutines are not used by any of the drivers. Some of these routines provide alternative methods for doing some of the computations and the others provide specialized capabilities not covered by the drivers.

In addition to the drivers there are 58 subroutines in EISPACK. This modularization greatly reduces both the amount of source and object code that must be handled. It also provides opportunities for using EISPACK capabilities in computations not envisioned during the original development. However, the user who desires access to these capabilities is faced with a formidable list of subroutines.

The efficient and accurate solution of a matrix eigenvalue problem usually involves at least two of the following steps:

Initial Scaling. An operation known as balancing that is applied to nonsymmetric matrices to reduce roundoff errors in subsequent calculations.

Reduction. Similarity transformation to a tridiagonal matrix in the symmetric case, or Hessenberg matrix in the nonsymmetric case (see 2.6).

Eigenvalue Iteration. Any of several iterative processes to compute the eigenvalues of the reduced matrix.

Eigenvector calculation. Any of several different methods to find eigenvectors of the reduced matrix.

Back transformation. Application of the inverse of the original reduction transformation to the matrix of eigenvectors.

Back scaling. Application of the inverse of the balancing transformation to the matrix of eigenvectors.

4.4 Choosing EISPACK Subroutines

For those specialized problems for which there are no drivers provided in EISPACK, the user has to select several subroutines, each of which performs one of the steps of the computational process as explained in the last section. A group of these subroutines is referred to as a path. The following tables list the EISPACK path for each combination of problem class (as discussed in chapter 2) and solution option (as discussed in chapter 3). Whenever appropriate, BCSLIB routines (*) and EISPACK drivers (+) are listed. The alternatives are denoted by numbers, the smallest number corresponding to the most preferable choice. If a problem/option combination does not appear in the tables, then this combination is either meaningless, or no appropriate routines are provided.

If EISPACK subroutines are accessed on BCS systems as explained in 5.2.2 and 5.3.2 the user obtains the single precision version on MAINSTREAM-EKS/VSP systems and the REAL*8 (long word) version on IBM MVS/TSO systems. These are the appropriate versions for scientific computations on the corresponding computers.

4.4.1 Standard Eigenvalue Problem for Real General Matrices

Solution option	Subroutines	Remarks
All eigenvalues	1)HSGEEV (HDGEEV)*	
	2)RG+	RG is equivalent to HSGEEV.
	3)BALANC	This path is equivalent to RG.
	ELMHES	
	HQR	
All eigenvalues and some eigen-vectors	BALANC	
	ELMHES	
	HQR	
	INVIT	
	ELMBAK	
	BALBAK	
All eigenvalues and all eigen-vectors	1)HSGEEV (HDGEEV)*	
	2)RG+	RG is equivalent to HSGEEV.
	3)BALANC	This path is equivalent to RG.
	ELMHES	
	ELTRAN	
	HQR2	
	ELMBAK	
	BALBAK	
Alternative I:	The subroutines ELMHES,ELTRAN,ELMBAK can be replaced by ORTRES,ORTRAN,ORTBAK	In general slower computation, but better numerical properties in some cases.
Alternative II:	BALANC and BALBAK may be omitted.	Only for well scaled problems, but not recommended.

4.4.2 Standard Eigenvalue Problem for Real Symmetric Matrices

Solution option	Subroutines	Remarks
All eigenvalues	1)HSSYEV (HDSYEV)*	
	2)RS+	RS is equivalent to HSSYEV.
	3)TRED1 TQL1	This path is equivalent to RS.
	4)TRED1 IMTQL1	Useful for higher accuracy in the smaller eigenvalues for matrices with wide variation in the size of the eigenvalues.
	5)TRED1 TQLRAT	Faster than TQL1, but requires one extra vector storage.
All eigenvalues and some eigen-vectors	TRED1 IMTQLV TINVIT TRBAK1	
All eigenvalues and all eigen-vectors	1)HSSYEV (HDSYEV)*	
	2)RS+	RS is equivalent to HSSYEV.
	3)TRED2 TQL2	This path is equivalent to RS.
	4)TRED2 IMTQL2	See 4) above.
The largest or smallest eigen-values	1)TRED1 TRIDIB	
	2)TRED1 RATQR	See 2.2.4 of [1]

The largest or smallest eigen-values and corres-ponding eigen vectors	1)TRED1 TRIDIB TINVIT TRBAK1 2)TRED1 RATQR TINVIT TRBAK1	See 2.2.4 of [1]
All eigenvalues in a specified interval	TRED1 BISECT	
All eigenvalues in a specified interval and corresponding eigenvectors	1)TRED1 BISECT TINVIT TRBAK1 2)TRED1 TSTURM TRBAK1	TSTURM uses one array more than BISECT, but less storage for integers.

4.4.3. Standard Eigenvalue Problem for Real Symmetric Packed Matrices.

Solution option	Subroutines	Remarks
All eigenvalues	1)RSP+	
	2)TRED3 TQL1	This path is equivalent to RSP.
	3)TRED3 IMTQL1	Useful for higher accuracy in the smaller eigenvalues for matrices with wide variation in the size of the eigenvalues.
	4)TRED3 TQLRAT	Faster than TQL1, but requires one extra vector storage.
All eigenvalues and some eigen-vectors	TRED3 IMTQLV TINVIT TRBAK3	
All eigenvalues and all eigen-vectors	1)RSP+	
	2)TRED3 TQL2	This path is equivalent to RSP.
	3)TRED3 IMTQL2	See 3) above.
All other options listed for real symmetric matrices	Use the corresponding paths for real symmetric matrices and replace TRED1 with TRED3, and TRBAK1 with TRBAK3.	See corresponding remarks for real symmetric matrices.

4.4.4. Standard Eigenvalue Problem for Real Symmetric Banded Matrices.

Solution option	Subroutines	Remarks
All eigenvalues	1)RSB$^+$	
	2)BANDR TQL1	This path is equivalent to RSB.
	3)BANDR IMTQL1	Useful for higher accuracy in the smaller eigenvalues for matrices with wide variation in the size of the eigenvalues.
	4)BANDR TQLRAT	Faster than TQL1, but requires one extra vector storage.
All eigenvalues and some eigen-vectors	BANDR IMTQLV TINVIT BANDV	Input matrix must be saved before call to BANDR for later use by BANDV.
All eigenvalues and all eigen-vectors	1)RSB$^+$	
	2)BANDR TQL2	This path is equivalent to RSB.
	3)BANDR IMTQL2	See 3) above.
All other options listed for real symmetric matrices	Use the corresponding paths for real symmetric matrices and replace TRED1 with BANDR, and TRBAK1 with BANDV.	See corresponding remarks for real symmetric matrices. See remark about BANDV above.

The eigenvalue closest to a shift T	BQR	BQR can be called repeatedly to find several different eigenvalues. BQR is less reliable than BANDR and BISECT, but it may be faster.
The eigenvalue closest to a shift T and corresponding eigenvector	BQR BANDV	See remarks above.

4.4.5 Standard Eigenvalue Problem for Real Symmetric Tridiagonal Matrices

Solution option	Subroutines	Remarks
All eigenvalues	1)RST+	
	2)TQL1	
	3)IMTQL1	Useful for higher accuracy in the smaller eigenvalues for matrices with wide variation in the size of the eigenvalues.
	4)TQLRAT	Faster than TQL1, but requires one extra vector storage.
All eigenvalues and some eigen-vectors	IMTQLV TINVIT	
All eigenvalues and all eigen-vectors	1)RST+	
	2)TQL2	
	3)IMTQL2	See 3) above.
All other options listed for real symmetric matrices	Use the corresponding paths for real symmetric matrices and omit all occurences of TRED1 and TRBAK1.	See corresponding remarks for real symmetric matrices.
The eigenvalue closest to a shift T with/without the corresp. vector	Use BQR or BQR+BANDV for this special case	See corresponding remarks in 4.4.4.

4.4.6 Standard Eigenvalue Problem for Real Sign-Symmetric Tridiagonal Matrices

Solution option	Subroutines	Remarks
All eigenvalues	1)RT$^+$	
	2)FIGI TQL1	This path is equivalent to RT.
	3)FIGI IMTQL1	Useful for higher accuracy in the smaller eigenvalues for matrices with wide variation in the size of the eigenvalues.
	4)FIGI TQLRAT	Faster than TQL1, but requires one extra vector storage.
All eigenvalues and some eigen-vectors	FIGI IMTQLV TINVIT BAKVEC	
All eigenvalues and all eigen-vectors	1)RT$^+$	
	2)FIGI2 TQL2	This path is equivalent to RT.
	3)FIGI2 IMTQL2	See 3) above.
All other options listed for real symmetric matrices	Use the corresponding paths for real symmetric matrices and replace TRED1 with FIGI, and TRBAK1 with BAKVEC.	See corresponding remarks for real symmetric matrices.

4.4.7 Standard Eigenvalue Problem for Complex General Matrices

Solution option	Subroutines	Remarks
All eigenvalues	1)HCGEEV (HZGEEV)*	
	2)CG$^+$	CG is equivalent to HCGEEV.
	3)CBAL COMHES COMLR	This path is equivalent to CG.
All eigenvalues and some eigen-vectors	CBAL COMHES COMLR CINVIT CBAK2	
All eigenvalues and all eigen-vectors	1)HCGEEV (HZGEEV)*	
	2)CG$^+$	CG is equivalent to HCGEEV.
	3)CBAL COMHES COMLR2 CBAK2	This path is equivalent to CG.
Alternative I:	The subroutines COMHES,COMLR,COMLR2, and COMBAK can be replaced by CORTH, COMQR,COMQR2, and CBAK2.	In general slower computation, but better numerical properties in some cases.
Alternative II:	CBAL and CBAK2 may be omitted.	Only for well scaled problems - not recommended.

4.4.8 Standard Eigenvalue Problem for Complex Hermitian Matrices

Solution option	Subroutines	Remarks
All eigenvalues	1)HCHIEV (HZHIEV)*	
	2)CH+	CH is equivalent to HCHIEV.
	3)HTRIDI TQL1	This path is equivalent to CH.
	4)HTRIDI IMTQL1	Useful for higher accuracy in the smaller eigenvalues for matrices with wide variation in the size of the eigenvalues.
	5)HTRIDI TQLRAT	Faster than TQL1, but requires one extra vector storage.
All eigenvalues and some eigen-vectors	HTRIDI IMTQLV TINVIT HTRIBAK	
All eigenvalues and all eigen-vectors	1)HCHIEV (HZHIEV)*	
	2)CH+	CH is equivalent to HCHIEV.
	3)HTRIDI TQL2 HTRIBK	This path is equivalent to CH.
	4)HTRIDI IMTQL2 HTRIBK	See 4) above.
All other options listed for real symmetric matrices	Use the corresponding path, but substitute HTRIDI for TRED1, and HTRIBK for TRBAK1.	See corresponding remarks in section 4.4.2.

4.4.9 Standard Eigenvalue Problem for Complex Hermitian Packed Matrices

Solution option	Subroutines	Remarks
All solution options listed for real symmetric packed matrices	Use the corresponding paths for real symm. packed matrices and replace TRED3 by HTRID3, and TRBAK3 by HTRIB3.	See the corresponding remarks in section 4.4.3. There is no EISPACK driver for this problem class.

4.4.10. Real Generalized Eigenvalue Problem.

Solution option	Subroutines	Remarks
All eigenvalues	1)HSGEGV (HDGEGV)*	
	2)RGG+	RGG is equivalent to HSGEGV.
	3)QZHES QZIT QZVAL	This path is equivalent to RGG.
All eigenvalues and eigenvectors	Same as above, but in path 3) add the subroutine QZVEC after QZVAL.	No paths for partial eigen-systems are recommended.

4.4.11 Generalized Real Symmetric Eigenvalue Problem

Solution option	Subroutines	Remarks
All eigenvalues	1)HSSYGV (HDSYGV)*	
	2)RSG^{+}	RSG is equivalent to HSSYGV
	3)REDUC TRED1 TQLRAT	This path is equivalent to RSG.
	4)REDUC TRED1 IMTQL1	See remarks about TQL1, IMTQL1, and TQLRAT in section 4.4.2.
	5)REDUC TRED1 TQL1	
All eigenvalues and some eigen-vectors	REDUC,TRED1,IMTQLV, TINVIT,TRBAK, REBAK	
All eigenvalues and all eigen-vectors	1)HSSYGV (HDSYGV)*	
	2)RSG^{+}	RSG is equivalent to HSSYGV.
	3)REDUC, TRED2,TQL2,REBAK	This path is equivalent to RSG.
	4)REDUC,TRED2,IMTQL2, REBAK	See remarks above.
All other options listed for real symmetric matrices	Use the corresponding path, and add REDUC at the beginning of each path. Add REBAK at the end for eigenvectors.	See corresponding remarks in section 4.4.2.

4.4.12 The AB Problem

Solution option	Subroutines	Remarks
All eigenvalues and all or none of eigenvectors	RSGAB$^+$	
All other options listed for the generalized real symmetric problem	Use the correspon-ding path from 4.4.11, but replace REDUC by REDUC2	See 4.4.11.

4.4.13. The BA Problem.

Solution option	Subroutines	Remarks
All eigenvalues and all or none of eigenvectors	RSGBA$^+$	
All other options listed for the generalized real symmetric problem	Use the correspon-ding path from 4.4.11, but replace REDUC by REDUC2, and REBAK by REBAKB.	See 4.4.11.

4.4.14 The Singular Value Decomposition

Solution option	Subroutine	Remarks
All singular values and vectors	SVD	For real general and real rectangular matrices only. For complex general matrices see subroutines in LINPACK [7].
Solution of linear least squares problems	MINFIT	See [2], section 2.4.2.

4.5 MATLAB

MATLAB is an interactive computer program available only on EKS CYBER computers. It serves as a convenient way to test computations involving matrices. In particular, MATLAB can provide quick solutions to eigenvalue problems for small matrices (less than 20 by 20). MATLAB provides easy access to the corresponding EISPACK subroutines. No Fortran driver program is needed.

To access and execute MATLAB, enter the EKS job control statements:

```
ATTACH,MATLAB/UN=EKSAPP,M=E.
MATLAB.
```

The HELP command in MATLAB gives information about the MATLAB capabilities. HELP INTRO will list an introduction to formatting conventions for matrices and vectors. Of interest for eigenvalue/vector calculations are the commands:

EIG(X)	produces a diagonal matrix D of eigenvalues of the matrix X, and a full matrix V of corresponding eigenvectors.
SVD(X)	singular value decomposition in matrices U,S, and V.

A file with the brief description of the symbols and commands of MATLAB can be printed with the following control statements:

```
GET,HELPMAT/UN=EKSAPP.
ROUTE,HELPMAT,DC=PR,PS,MB=mailbox,UN=RJE user number.
```

For more information see the Mainstream EKS Application Memo 60-006, the Newsletter article on MATLAB (Vol.7 No.1), or contact the consultants (see section 1.3.2).

4.6 The EISPACK Control Program EISPAC

The EISPACK control program EISPAC is available on the TSO system. EISPAC is designed to simplify the solution of eigenvalue problems with EISPACK. The user describes a problem to EISPAC in simple familiar terms. EISPAC then chooses the appropriate EISPACK path automatically and executes the subroutines in proper order, passing parameters from one subroutine to the next in the proper way. EISPAC has most, but not all of the capabilities provided by the individual EISPACK subroutines. However, EISPAC relieves the user of having to develop a detailed knowledge of the applicability, efficiency, and calling sequences of the individual EISPACK subroutines.

More information about EISPAC is provided in [1] and [2]. To obtain usage documentation for EISPAC follow the directions in section 5.3.1 and use the name EISPACDC in the name card. The EISPAC control program is obtained from the EISPACK library as shown in section 5.3.2.

5.0 How to Access EISPACK

5.1 General Remarks

Subprograms in BCSLIB do not require any special access instructions, since BCSLIB is the default library on any BCS system (EKS, TSO, VAX, and PDP). To access the BCSLIB subroutines it is sufficient to include the corresponding CALL statements in the user's FORTRAN code.

5.2 MAINSTREAM-EKS/VSP Cyber and Cray Systems

5.2.1 Accessing EISPACK Subprogram Documentation

EISPACK subprogram documentation is contained in an UPDATE program library which is available on tape. The folowing job accesses the tape and extracts the documentation for individual subprograms.

```
DOCJOB.
USER,usernum,passwrd. name/phone/location
LABEL,T,SI=MTHPROD,UN=EKSAPP.
COPYBF,T,OLDPL.
RETURN,T.
UPDATE,Q,D,L=1,Y.*C,sublist
COPY,COMPILE,OUTPUT.
```

Here sublist is the list of names of those subprograms whose documentation is desired. The names are separated by commas. For example, to obtain documentation for subprograms RG and CG use the UPDATE statement

```
UPDATE,Q,D,L=1,Y.*C,RG,CG
```

If all the names will not fit on one UPDATE statement (80 characters max), the pair of statements

```
UPDATE,Q,D,L=1,Y.*C,sublist
COPY,COMPILE,OUTPUT.
```

may be repeated as often as needed with a new sublist.

If all EISPACK documentation is desired (approximately 500 pages), use the UPDATE statement

```
UPDATE,Q,D,L=1,Y.*C,BAKVEC.TSTURM
```

5.2.2 Accessing EISPACK as a Subprogram Library

EISPACK is available in both FTN4 and FTN5 compatible libraries on the Cyber system and in a CFT compatible library on the Cray system. These libraries may be accessed as follows:

```
          FTN4 library: ATTACH,EISLIBF/UN=EKSAPP.
          FTN5 library: ATTACH,EKSLIB5/UN=EKSAPP.
          CFT  library: ACCESS,DN=EISLIBC,UN=EKSAPP.
```

Use of the EISPACK library requires its mention on a loader control statement in addition to one of the above library access statements. On the Cyber system, this can be done with the U parameter on the LOADXEQ statement or the LIB parameter on the LDSET statement. On the Cray system, it can be done with the LIB parameter on the LDR statement. The subroutines accessed in this way are single precision.

5.3 IBM MVS/TSO Systems

5.3.1 Accessing Documentation for a Specific Subroutine

EISPACK subroutine usage documentation can be accessed with the following JCL and control cards:

```
          //  EXEC MATHLIST
          //SYSIN  DD  *
          SELECT                              (select card)
          Sn1 Sn2 ...                         (name card,columns 1-72 only)
```

where SELECT is in columns 1-6. Sn_1 Sn_2 ... are the names of the subroutines whose usage documentation is desired, with DC appended on the end of each name, and the names separated by one or more blanks. For example, use BISECTDC to obtain documentation for subroutine BISECT. Use as many cards as needed (columns 73-80 are ignored).

To list all the EISPACK subroutine documentation, use only a select card with the word ALL in columns 1-3 in place of the word SELECT, and do not use any name cards. Use ALL with discretion. The complete documentation has about 500 pages.

5.3.2 Accessing EISPACK as a Subroutine Library

The EISPACK library may be used with Fortran programs compiled by any of the IBM standard or program product Fortran compilers. For example, the EISPACK library would be used with the FORTRAN H extended compiler as follows:

```
// EXEC FORTXCLG,LIB3='ENG.MATH.EISPACK2'
//FORT.SYSIN  DD  *
     the FORTRAN source deck
//GO.SYSIN  DD  *
     the input data
```

The significant point in this example is adding

```
,LIB3='ENG.MATH.EISPACK2'
```

to the EXEC card.

Two or more Math/Stat Speciality Libraries may be accessed jointly as follows:

```
// EXEC FORTXCLG
//FORT.SYSIN  DD  *
     the FORTRAN source deck
//LKED.SYSLIB  DD
// DD
// DD  DISP=SHR,DSN=ENG.MATH.EISPACK2
// DD  DISP=SHR,DSN=ENG.MATH.LEVELTWO
//GO.SYSIN  DD  *
     the input data
```

The subroutines accessed in this way expect all real variables to be of type REAL*8.

If the EISPAC control program is used the sequence is:

```
// EXEC FORTCLG,LIB3='ENG.MATH.EISPACK2'
//FORT.SYSIN  DD  *
     the FORTRAN source deck
//GO.EISPACLB  DD  DSN=ENG.MATH.EISPACK2,DISP=SHR
//GO.SYSIN  DD   *
```

6.0 Returns from EISPACK

6.1 A Frequent Source of Problems

A frequent source of problems for users of linear algebra software including EISPACK is the way doubly dimensioned arrays are handled by FORTRAN subroutines. The majority of the problems reported by EISPACK users can be traced to either incorrectly dimensioned arrays in the calling program, or incorrect use of the "row dimension" parameter in the calling sequence of EISPACK routines. Users who encounter this problem should consult section 1.4.3. in the BCSLIB manual, or the short note on "'Row Dimension' in Matrix Algebra Software" in the Newsletter (Vol.2 No.2).

6.2. Error Returns from EISPACK.

A summary of error returns from EISPACK subroutines is given in the following table (from [2]). Note that for certain nonzero values of IERR some meaningful results may have been obtained. More information on the particular errors can be found in the documentation of the subroutine in question.

IERR	SUBROUTINES	EISPAC MESSAGE	SIGNIFICANCE OF THE ERROR
i $1 \leq i \leq n$	MINFIT,QZIT, RGG,RSB,RSG, RSGAB,RSGBA, SVD,TQLRAT, TQL2	00	The calculation of the i-th eigenvalue or singular value failed to converge. If MINFIT,QZIT (with QZVAL),RGG, or SVD was being used, the eigenvalues or singular values i+1,i+2,...N should be correct; otherwise the eigenvalues 1,2,..i-1 should be correct. No eigenvectors are correct.

IERR	SUBROUTINES	EISPAC MESSAGE	SIGNIFICANCE OF THE ERROR
N	BQR	--	The calculation of the eigenvalue failed to converge.
3N+1	BISECT	03	The parameter MM specified insufficient storage to hold all the eigenvalues in the interval (RLB,RUB). The only useful result is M, which is set to the number of eigenvalues in the interval.
7N+1	REDUC,REDUC2, RSG,RSGAB, RSGBA	07	The matrix BR is not positive definite as required in the real symmetric generalized paths. No useful results are produced.
-i $1 \leq i \leq N$	BANDV,TINVIT	50	The calculation of one or more of the eigenvectors including the i-th vector failed to converge; these vectors are set to zero. These failures may be caused by insufficient accuracy in the corresponding eigenvalues. All non-zero eigenvectors and their corresponding eigenvalues should be correct.
10N	RGG,RSB,RSG, RSGAB,RSGBA	--	The parameter N specifying the order of the input matrix or system exceeds the dimension parameter NM. No results are produced.
12N	RSB	--	Either the parameter MB specifying the (half) band width is non-positive or it exceeds the matrix order N. No results are produced.

6.3 The Accuracy of the Computed Results

(adapted from [5])

Effective use of the EISPACK subroutines demands an understanding of the accuracy one can expect from the computed results. Various sources of errors may enter the computation:

- 0 The matrices may consist of, or are derived from inexact data.
- 0 The computation with the fixed precision floating point arithmetic of the host computer introduces roundoff errors.

Even if these sources of error did not exist, it would not be possible to compute eigenvalues of general matrices exactly in a finite number of steps. It can be shown that no such mathematical algorithm exists.

The best that can be said about the algorithms used by EISPACK to compute eigensystems is that they are <u>numerically stable</u>. This means that EISPACK produces the exact answer to an eigenvalue problem involving a matrix A+E which is a small perturbation of the given matrix A, i.e. the norm of the perturbation matrix E is roughly the size of roundoff error when compared to the norm of A. The same can be said for the algorithms based on non-orthogonal transformations if no exceptional growth in the size of the matrix elements occurs during the computation.

One immediate consequence of this numerical stability is that the computed results will produce small residuals. If λ and $\mathbf{x}$ are a computed eigenvalue and eigenvector of a given matrix A, then $A\mathbf{x}$ will always be close to $\lambda\mathbf{x}$. More precisely the size of the relative residual,

$$\frac{||A\mathbf{x} - \lambda\mathbf{x}||}{||A|| \; ||\mathbf{x}||}$$

can always be expected to be roughly equal to the relative accuracy of floating point arithmetic on the computer being used.

But how close are the computed eigenvalues to the exact eigenvalues? The answer to this question depends more on the matrix involved than on the particular EISPACK subroutine used to do the computation. To see why this is so, assume that A has a complete, linearly independent set of eigenvectors X and let D denote the diagonal matrix of eigenvalues. Then

$$X^{-1}AX = D.$$

Suppose that A is perturbed somehow, either by errors in its initial formation or by roundoff errors generated by EISPACK. Then

$$X^{-1}(A + E)X = D + X^{-1}EX.$$

The resulting perturbation to D is not diagonal, but this equation makes it plausible that the damage done by E to the eigenvalues in D could be as large as $||X^{-1}||\ ||E||\ ||X||$, rather than merely $||E||$. The quantity

$$k(X) = ||X||\ ||X^{-1}||,$$

the <u>condition</u> <u>number</u> of X, arises in the perturbation analysis for systems of linear equations. Note that in the eigenvalue problem it is the condition of X, the matrix of eigenvectors, not of A itself, that is relevant. If A is real and symmetric, then X can be taken to be orthogonal and k(X) with respect to the 2-norm) is 1. For such matrices A, a small change in the matrix causes a small change in the eigenvalues. In other words the eigenvalues of symmetric matrices are always well conditioned.

In the extreme case where A does not have a full set of eigenvectors, k(X) should be regarded as infinite. The eigenvalues are infinitely sensitive to perturbations in the matrix.

With EISPACK the consequences of this perturbation theory are the following:

For real symmetric matrices, and for complex Hermitian matrices, the eigenvalues are always computed with an accuracy that corresponds to a few units of roundoff error in the largest eigenvalue of the matrix. The small eigenvalues of the symmetric matrices will not necessarily be computed to full accuracy relative to the norm of the matrix. For symmetric matrices the presence of multiple eigenvalues has little effect on the accuracy of the computed results.

For general, nonsymmetric matrices, the effect of the roundoff errors on the computed eigenvalues will increase as the condition number of the eigenvector matrix increases. If a nonsymmetric matrix has multiple eigenvalues, or is close to a matrix with multiple eigenvalues, and its eigenvector matrix has a large condition number, then the computed eigenvalues may be accurate to less than full precision.

As an example consider the following matrix.

$$\begin{bmatrix} -64 & 82 & 21 \\ 144 & -178 & -46 \\ -771 & 962 & 248 \end{bmatrix}$$

This matrix was constructed in such a way that the exact eigenvalues happen to be 1, 2, and 3. When the eigenvalues are computed using EISPACK subroutines on a computer with 24-bit floating point arithmetic, the results are

1.00195
2.00113
2.99736

Such a computer has a relative floating point accuracy of better than 10^{-6}, so the computed eigenvalues have lost half the available figures.

The difficulties lie with the matrix itself, not with EISPACK. The matrix of computed eigenvectors, renormalized so that the last component of each vector is one, is

$$X = \begin{bmatrix} 0.090922 & 0.075114 & 0.111016 \\ -0.181810 & -0.196554 & -0.166741 \\ 1.000000 & 1.000000 & 1.000000 \end{bmatrix}$$

The condition number of X is greater than 10^{-3}. A single roundoff error in A, which on this computer affects the 6-th significant figure, could cause changes in the 3-rd significant figure of the eigenvalues. EISPACK has computed the eigenvalues as accurately as possible using floating point arithmetic of this accuracy.

As another measure of accuracy, let D be the diagonal matrix whose diagonal elements are the computed eigenvalues. Then

$$\frac{||AX - XD||}{||A||\ ||X||} = 0.13{*}10^{-6}$$

In other words, the relative residuals are on the order of roundoff error, even though the eigenvalues are accurate to only three figures. For further information on the perturbation theory for the eigenvalue problem, see [8].

Appendix A. Programmed Instructions for Using the EISPACK Guide.

STEP 1 : IDENTIFIY YOUR EIGENVALUE PROBLEM

Instructions: Read sections 2.1, 2.2, and 2.3 and decide to which of the following problem classes your eigenvalue problem belongs:

Real General
Real Symmetric
Real Symmetric Packed
Real Symmetric Tridiagonal
Real Sign-Symmetric Tridiagonal

Complex General
Complex Hermitian
Complex Hermitian Packed

Generalized Real
Generalized Symmetric

AB - Problem
BA - Problem

Singular Value Decompositon

If your problem does not fall into any of the classes listed above, or if you have difficulties in classifying the problem, call the consultants (see section 1.3.2).

STEP 2: DETERMINE YOUR COMPUTATIONAL TASK

Instructions: Read Chapter 3 and decide which of the following solution options you want to choose:

A) All eigenvalues

B) All eigenvalues and some selected eigenvectors

C) All eigenvalues and all eigenvectors

D) The largest or smallest eigenvalues

E) The largest or smallest eigenvalues and the corresponding eigenvectors

F) The eigenvalues in a specified interval

G) The eigenvalues in a specified interval and the corresponding eigenvectors.

Note that the solution options D-G apply only to symmetric (Hermitian, generalized symmetric) eigenproblems.
There are no options to choose from for the singular value decomposition.

STEP 3: FIND THE APPROPRIATE SUBROUTINE(S)

Instructions: Read sections 4.1 and 4.2. Then use the tables in section 4.4 to find the appropriate subroutine(s) for the given problem/solution option combination.

If no subroutines are listed for the given combination, then go back to STEP 2 and choose a more general solution option, or call the consultants (see section 1.3.2).

If several possibilities are listed for the given combination, then choose the subroutine(s) listed first.

STEP 4 : OBTAIN ACCESS INFORMATION AND DOCUMENTATION

Instructions: Read Chapter 5 and determine how the desired subroutines are accessed on the computer system you are using.

If you have problems accessing EISPACK, or if the desired subroutines are not available, call the consultants (see section 1.3.2).

If needed, obtain documentation on the EISPACK subroutines as described in Chapter 5.

STEP 5 : WRITE YOUR PROGRAM USING THE SELECTED SUBROUTINES AND TEST IT

If you have any problems, consider the remarks in Chapter 6, or call the consultants (see section 1.3.2).

Use the computed results for your purposes, keeping the remarks in section 6.3 in mind.

Appendix B. References.

[1] B.T. Smith et.al. , Matrix Eigensystem Routines - EISPACK Guide, Springer Lecture Notes in Computer Science 6, 1974.

[2] B.S. Garbow et.al. , Matrix Eigensystem Routines - EISPACK Guide Extension, Springer Lecture Notes in Computer Science 51, 1977.

[3] a) TSO/CTS BCSLIB Math/Stat/Utility Subprogram Library, Users Manual, 10023-R2, Boeing Computer Services Company, 1982.
b) MAINSTREAM-EKS/VSP BCSLIB Math/Stat/Utility Subprogram Library, Users Manual, 10208-156, Boeing Computer Services Company, 1983.

[4] Math/Stat Software Newsletter, Boeing Computer Services Company 1976 - 1983.

[5] J.J. Dongarra and C.B. Moler, EISPACK - A Package for Solving Matrix Eigenvalue Problems, in "Sources and Development of Mathematical Software", edited by W. Coswell, Prentice Hall, 1983.

[6] J.W. Wilkinson and C. Reinsch, Handbook for Automatic Computation, Vol. 2: Linear Algebra, Springer Verlag, Berlin 1971.

[7] B.N. Parlett, The Symmetric Eigenvalue Problem, Prentice Hall, Englewood Cliffs, 1980.

[8] J.H. Wilkinson, The Algebraic Eigenvalue Problem, Clarendon Press, Oxford, 1965.